3.3.6. SORTING ROUTINE.

3.3.6.1. INTRODUCTION. The sorting routine sorts a list, stored as array A, into ascending order.

3.3.6.2. INPUTS. The sorting routine has to be supplied with the appropriate numerical values of the following parameters:

- beginaddress of the list A, i.e. ii
- endaddress of the list A, i.e. jj
- desired ordering, i.e. ascending or descending
- length of the elements of A
- the relative position of the reference elements
- q

in case q has to be adapted:

- previous sorting time
- sign of change of q on previous run.

3.3.6.3. PROCESSING. The procedure sorts the elements of the list A [ii : jj] into ascending order. It continually splits the list into parts, such that all elements of one part are less than all elements of the other, with a third part in the middle consisting of a single element. An element with value t is chosen as pivot element. i and j give the lower and upper limits of the segment being split. After the split has taken place a value p will have been found such that A [p] = t and A [I] $\leq$ t $\leq$ A [J] for all I, J such that i $\leq$ I $\leq$ p $<$ J $\leq$ j. The program then performs operations on the two segments A [i : p - 1] and A [p + 1 : j] as follows: the smaller segment is split and the position of the larger segment is stored in the IL and IU arrays (IL and IU are mnemonics for Index Lower and Index Upper). Their demension is $\log$ (jj - ii + 2). If the segment to be split has q or fewer elements it is sorted directly, instead and another segment is obtained from the IL and IU arrays. When no more segments remain, the list A [ii:jj] is completely sorted. The optimal value $\hat{q}$ of the parameter q depends on the initial ordering present in the list A [ii:jj] , $\hat{q}$ being higher if the initial ordering is better. For random lists, a value of 6 or 10 for q will prove to be optimal. For non random lists, q can be adapted by increasing it depending upon part sorting times.
The optimal value of t, the value of the pivot element, is the median of the list being split. Unfortunately, this value is known only after completion of the sorting process. Therefore, an estimate of the median has to be made. Satisfactory performance is obtained by estimating t from the median of a sample of three elements of A. These elements are the first, middle and last element of the list A.
For random lists, the method of Het Mathematisch Centrum, which uses two pivot elements instead of one, is faster, but for non random lists it is not faster. Therefore, it is not discussed here. It is a simple matter to change the sorting routine into one that operates on equidistant memory elements instead of on successive memory elements. Also, the sorting routine can easily be generalized into one that sorts either into ascending or descending order.

3.3.6.3. (continued)

In case the sorting routine has to sort lists (tracktables for instance) according to the numerical value of one of their elements, (called the reference element) instead of sorting single elements, it becomes inefficient to exchange two lists completely, when during the sorting process two elements have to be exchanged. In this case, the original lists to be sorted are replaced by lists with only two elements: one element representing the numerical value according to which the sorting is to take place, and one representing the location (starting address) of the original lists. When the sorting of the two-element lists is complete, the original lists can be placed in their proper place, if desired, thus minimizing the number of exchanges.
The splitting of the list into parts such that all elements of one part are less than all elements of the other, proceeds as follows: As pivot element the median value of the first, middle and last element is chosen. Further more, these three elements are placed in ascending order. Then, starting from the next-to-last element the list is searched upward for an element less than or equal to the pivot element. When such an element A [L] has been found, the list is searched downward starting from the second element for an element greater than or equal to the pivot element. When such an element A[k] has been found, A[k] and A[L] are exchanged, provided that K≦L, and the search for elements to be exchanged is repeated in the list A[K+1 : L-1]. Because the middle element is equal to the pivot element, automatic stopping is assured.

3.3.6.4. OUTPUTS.

- the sorting time
- value of q
- sign of change of q
- beginaddress of the list A
- endaddress of the list A

3.3.6.5. SPECIAL REQUIREMENTS. When it is required to sort a list with a good initial ordering, it is advantageous to have an adaptive sorting routine. Therefore, the sorting routine shall be adaptive, i.e. q shall remain constant, or be increased or decreased by one, depending upon the previous and present sorting time, and the sign of change of q on the previous run.
The inital value of q should be chosen between 6 and 10.
For short lists (≦40 elements) it is felt that adaptation will not be efficient, because the possible gain in sorting time will be balanced by the time needed for adaptation.

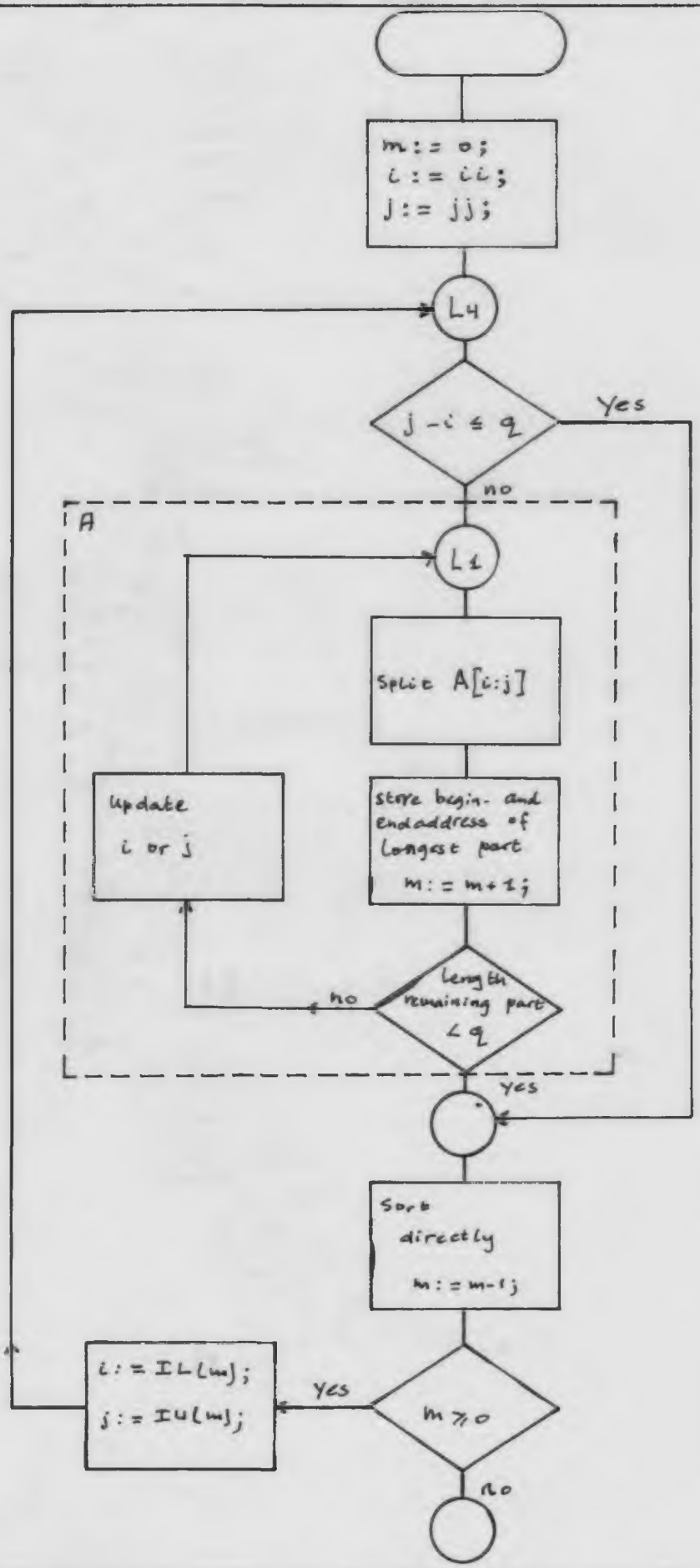
MINISTERIE VAN DEFENSIE (marine)
Centrum Automatisering Wapen- en Commandosystemen (CAWCS)

Opst.:
Acc. :
Datum voorg. uitgifte:

```
procedure SORT(A,ii,jj,q,w);
array A; integer ii,jj,q,w;
begin real x,z,xx,zz,y,t,tt; integer p,k,L,m,i,j; integer array IL,IU[0:w];
      switch ss:= L1,L2,L3,L4,L5;
      m:= 0; i:= ii; j:= jj; goto L4;
L1:   p:= (i+j) div 2; t:= A[p]; k:=i ; L:= j;
      x:= A[i];
      if x>t then
      begin
            y:=t; A[p]:=t:=x; A[i]:=x:=y
      end ;
      z:= A[j];
      if z<t then
      begin
            y:=t; A[p]:=t:=z; A[j]:=z:=y;
            if x>t then
            begin
                  y:=t;A[p]:=t:=x; A[i]:=x:=y
            end
      end;
L2:   L:=L-1;
      zz:=A[L];
      if zz>t then goto L2;
L3:   k:=k+1;
      xx:=A[k];
      if xx<t then goto L3;
      if k≤ L then
      begin
            y:=zz; A[L]:=zz:=xx; A[k]:=xx:=y;
            goto L2
      end;  if L-i > j-k then
      begin IL[m]:= i; IU[m]:= L; i:=k end
      else
      begin IL[m]:= k; IU[m]:= j; j:= L ; end ;
      m:= m+1;
L4:   if j-i > q then goto L1;
      for L:= i+1 step 1 until j do
      begin
            x:=A[L];
            k:= L-1;
            xx:= A[k];
            if xx>x then
            begin
L5:               A[k+1]:=xx;
                  k:=k-1;
                  if k≥ i then
                  begin
                        xx:=A[k];
                        if xx>x then goto L5
                  end;
                  A[k+1]:= x
            end
      end;
      m:= m-1; if m ≥ 0 then
      begin i:=IL[m]; j:=IU[m]; goto L4 end
end SORT;
```

MINISTERIE VAN DEFENSIE (marine)
Centrum Automatisering Wapen- en Commandosystemen (CAWCS)

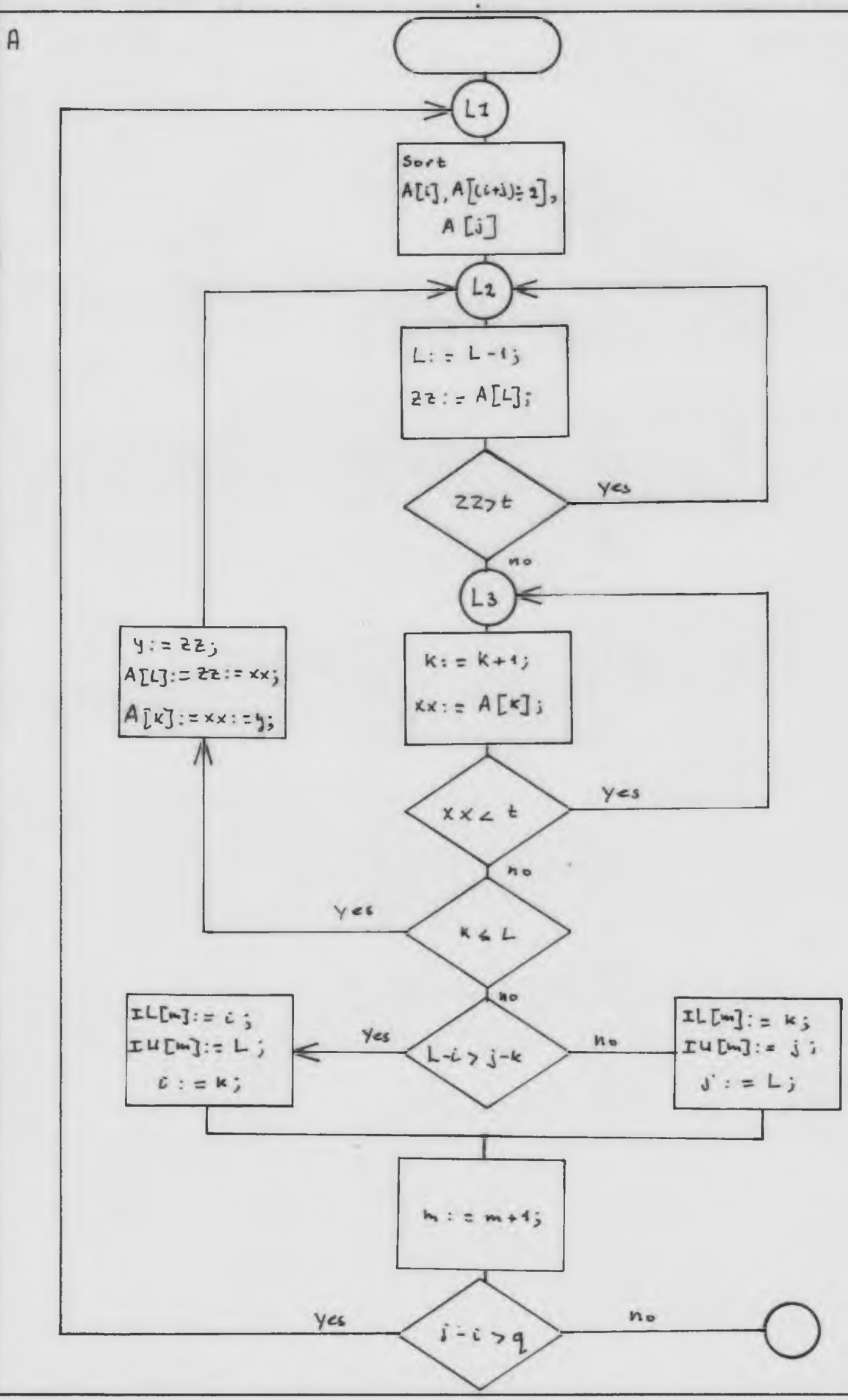
Opst.:
Acc. :
Datum voorg. uitgifte:



MINISTERIE VAN DEFENSIE (marine)
Centrum Automatisering Wapen- en Commandosystemen (CAWCS)

Nr. :
Hfdst :
Blad :
Datum:

Opst.:
Acc. :
Datum voorg. uitgifte:

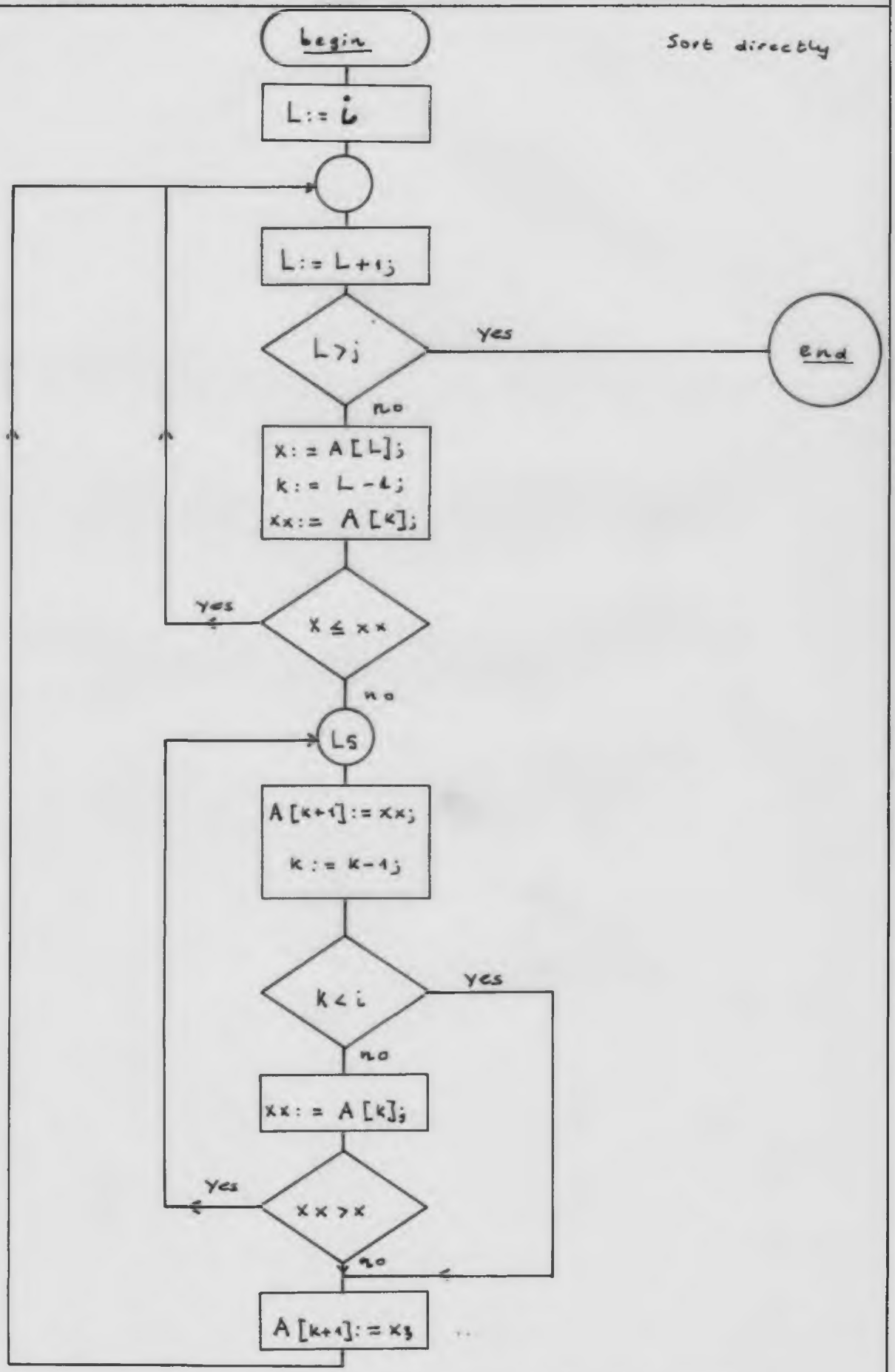


MINISTERIE VAN DEFENSIE (marine)
Centrum Automatisering Wapen- en Commandosystemen (CAWCS)

943183*-43

Nr. :
Hfdst :
Blad :
Datum:

Opst.:
Acc. :
Datum voorg. uitgifte:



MINISTERIE VAN DEFENSIE (marine)
Centrum Automatisering Wapen- en Commandosystemen (CAWCS)

943183*-43